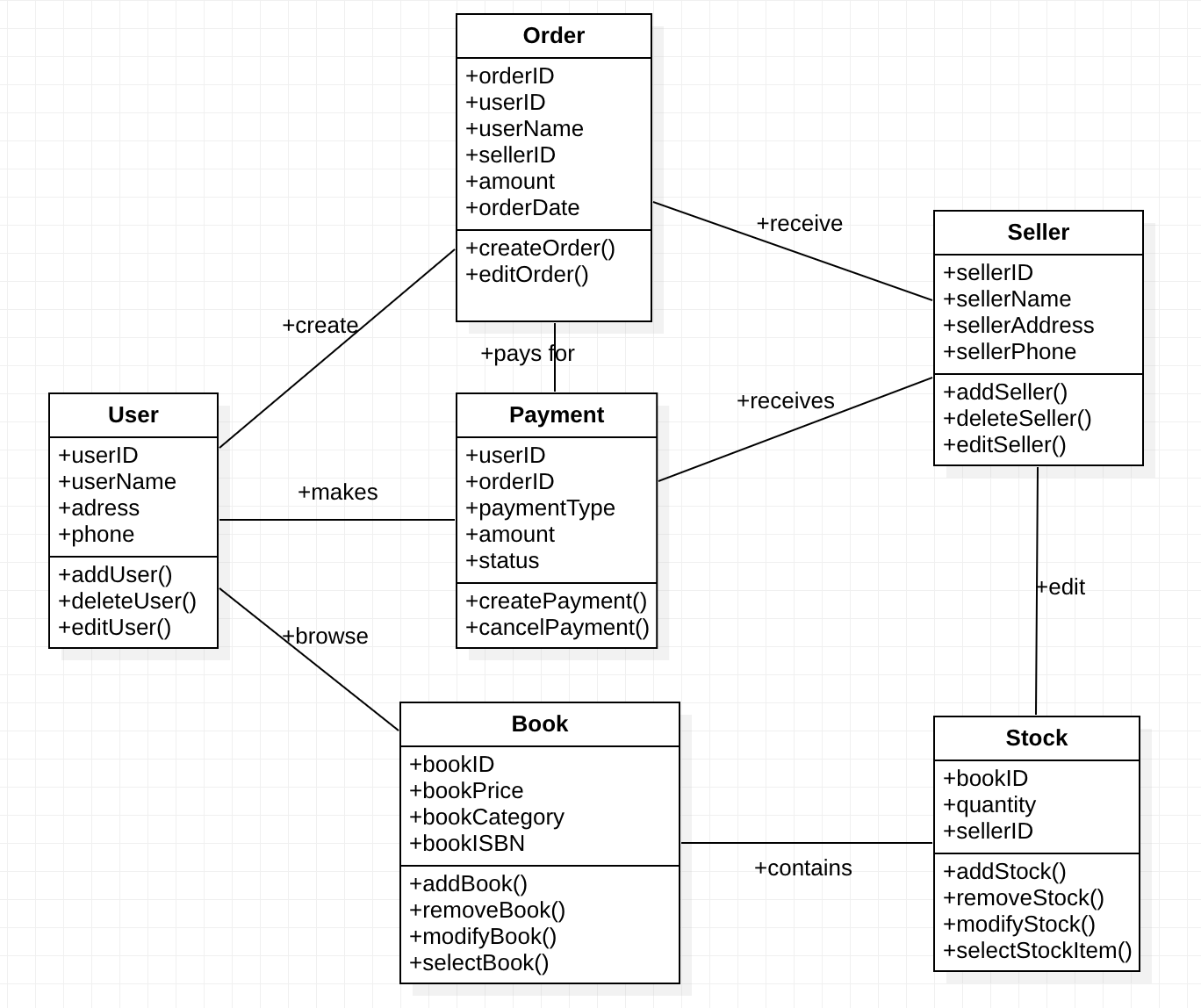
**Task 3 – Creating a Class diagram and design pattern selection**

**A class diagram containing classes and associations.**



**Class diagram functions:**

1. User creates order and received by seller.
2. User makes payment and received by seller.
3. User browses books which contain the stock as edited by seller.
4. Seller receives payment from user.
5. Seller receives order created by user.

**UML diagram representing the design pattern.**

The factory pattern is one of the most common design patterns in Java. This type of design pattern is classified as a creational pattern because it provides one of the best ways to create an object.

We use the Factory pattern to generate objects without revealing the creation mechanism to the client, and we refer to freshly formed objects using a common interface.

It's really simple to handle. When creating a new storage class, simply place it in the same package as other comparable classes, register it in the static block, and you're done.

The Factory Method Pattern allows sub-classes to specify the type of object to be created. It encourages loose coupling by removing the requirement to connect application-specific classes into code. That is, the code only interacts with the resulting interface or abstract class, thus it will function with any classes that implement that interface or extend that abstract class.

Another advantage of the factory design is that it requires less code changes and no if else statements in the factory classes. In fact, when you add a new storage implementation, you don't need to modify anything in the factory class. It also adheres to all good programming principles.